EMWIN-N Document 5

EMWIN Library

User's Guide

Avtec Systems, Inc.

Date Prepared: December 22, 2003

TM-03-059-00

AVTEC SYSTEMS

EMWIN Library

User's Guide

The information in this document has been fully reviewed and is believed to be entirely reliable. Avtec reserves the right, however, to modify any products herein to improve reliability, function, or design. Avtec does not assume any liability arising out of the application or use of any product or circuit described herein. Avtec does not convey any license under its patent rights or the rights of others.

TABLE OF CONTENTS

1. INTRODUCTION

The Emergency Manager's Weather Information Network (EMWIN) distributes various NWS products such as forecasts, watches, and warnings through various distribution media. A primary medium is via satellite distribution, often from the GOES satellites. As part of the upgrade to the next generation of GOES satellites, modulation and channel coding changes are being instrumented for the EMWIN downlink signal. In order to facilitate this upgrade and provide backwards compatibility for the existing modulation, this library, in the form of a Windows Dynamic Link Library (DLL), has been created to perform the demodulation and basic processing in software. The output of this library may then be directly sent to existing EMWIN processing software using the serial port, or accessed via TCP/IP.

This document is designed to provide information on how the library is to be used, as well as where and how it may fit into an EMWIN receiving station architecture. As such, it is assumed the reader knows a moderate amount about the EMWIN service, as well as both the EMWIN-I DFSK signal and the new generation EMWIN-N BPSK signal. Since this manual deals primarily with a software library, it also assumes some knowledge of software programming and TCP/IP interfaces.

Before delving to all the details of the library, the configuration, use, etc., it is useful to examine the general flow of the signal and data, both in hardware and software, to understand where the library fits into the receive station architecture.

1.1. HARDWARE DATA FLOW

As this library performs software demodulation, the raw modulated signal first must be ingested into a form the library can interpret. The following diagram gives an overview of how this is accomplished:

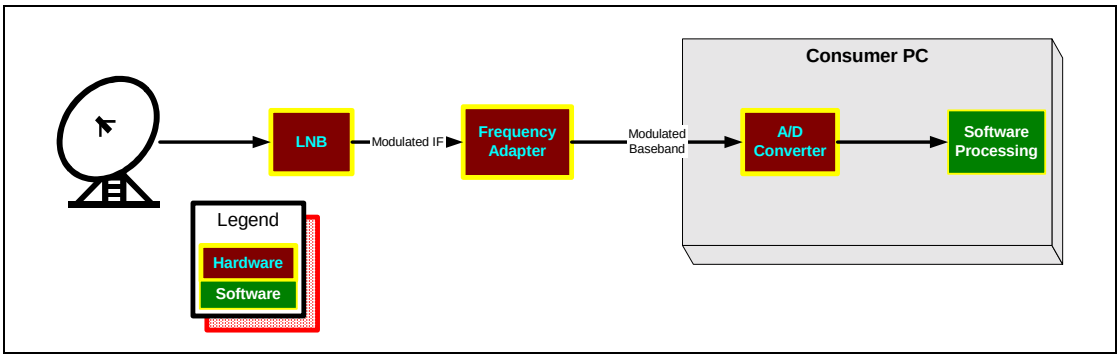

Figure 1-1: Hardware Data Flow Diagram

Once the antenna receives the downlinked signal, it is sent to the LNB, which then downconverts the received signal to an IF. This modulated IF signal is then sent to the Frequency Adapter. The purpose of the Frequency Adapter to take the IF signal and condition it so that it can be sampled by the A/D Converter. The A/D Converter is the final step performed in hardware.

1.2. SOFTWARE DATA FLOW

Once the signal is sampled by the A/D Converter, it can then go through the demodulation and data processing. The following diagram gives an overview of the processing that is performed:

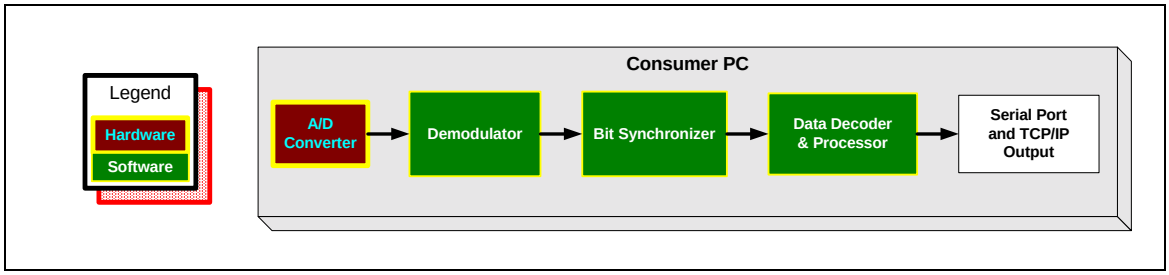

Figure 1-2: Software Data Flow Diagram

After the data is sampled by the A/D Converter, the signal is demodulated according to the selected format, EMWIN-I or EMWIN-N. It is then passed to the Bit Synchronizer, and then the Decoder/Processor. After all processing has been completed, the data is then output to both a serial port and a TCP/IP socket. The serial port allows easy integration in with existing EMWIN software using a simple null-modem cable. The TCP/IP port, operating as a server, allows support for future, more network-oriented systems.

2. INSTALLATION

The installation process for the library has been designed to be both easy and flexible. As such, it is provided as an archive of files. Included in the archive are the following files and directories:

DLLTester/	Contains an example program for the DLL
Data Source Service.bat	Script file for running the Data Source Service
Data Source Service.exe	Data Source Service Executable
DisplaySoundInterface.exe	Show the device number and names of available sound devices
EMWINSettings.reg	Registry entries for the default library settings
Lib/	Contains the DLL and export library
include/	Contains the C++ header files for the configuration and status structures, as well as the function definitions

System Requirements:

Operating System:	Microsoft Windows NT/2000/XP
Processor:	1.2 GHz or higher
Memory:	512 MB or higher
Disk Space:	2 MB for the library and associated software

To install the library, unzip the archive into a desired directory. Since the registry must be incorporated into that of the system, merge them into the system registry by double-clicking on the file using Explorer. The user should then be ready to start developing their application.

3. DATA SOURCE SERVICE

The Data Source Service operates as the bridge between the receiving hardware, antenna, LNB, etc., and the software demodulator and processing. In order to provide the flexibility to use different types and brands of A/D converters, the Data Source Service functionality exist outside of the of the EMWIN library. This allows the EMWIN library to be used as-is with different A/D converters, simply by swapping out a different Data Source Service, which implements the standard configuration and exchange interface.

The provided implementation of the Data Source Service is designed to work with the M-Audio Audiophile 2496 sound card. This sound card, used as the A/D Converter, is capable of the 96 Ksps sample rate, as well as having the greater than 40 KHz frequency response.

The Data Source Service, normally operating as a Windows Service, performs its functions as a transactional broker between the A/D converter and the EMWIN library. The service packs the sampled signal into messages structures defined in *include/SMsgHeader.h*. Once packed, the message is sent to the library, which acts as a TCP/IP client.

The service, operating as a TCP/IP server, is controlled by a TCP/IP client, the library, through command messaging. The service only sends data and response messages to the client. No response messages are sent for successful or failed data transmissions. The sequence for controlling the server is defined by the following diagram:

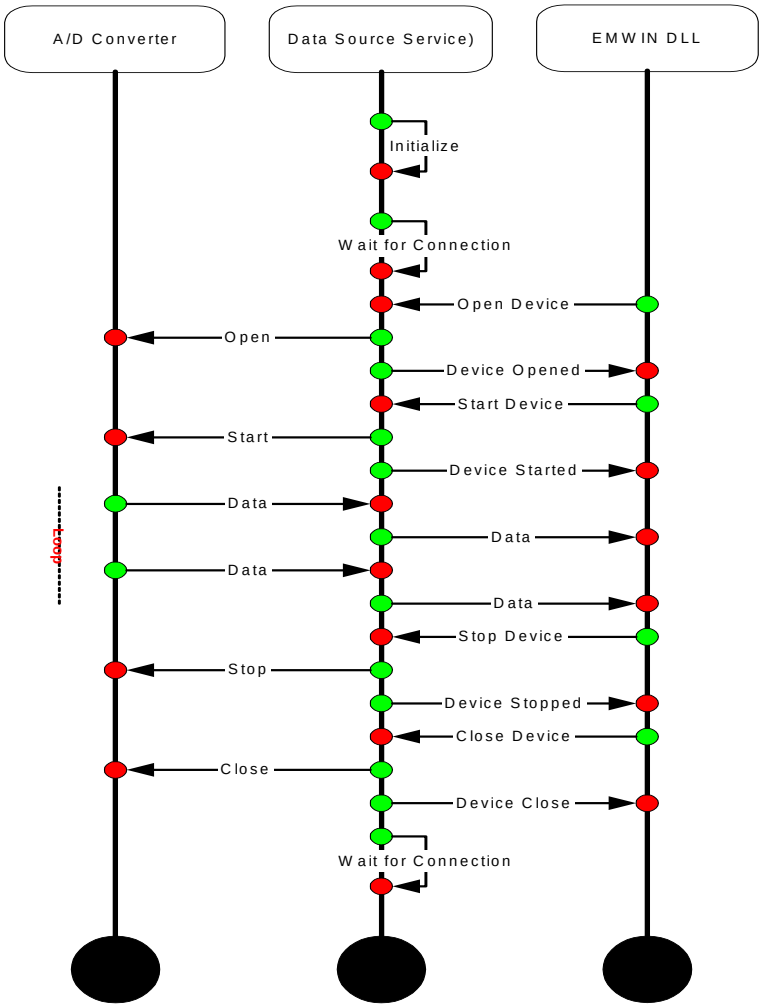

Figure 3-1: Data Source Service Diagram

The controlling mechanism are messages, sent from either the client or server to the other party. The messages have the following structure:

[Identifier][Time Issued][Expiration][Device Number][Sequence Number][Body Length]

where:

Table 3-1: Controlling Mechanism Table

MESSAGE FORMAT	DATA TYPE	DESCRIPTION
Identifier	unsigned short	Defines message type
Time Issued	unsigned short	Future
Expiration	unsigned char	Future
Device Number	unsigned char	Future
Sequence Number	unsigned short	Message order number
Body Length	unsigned short	Length of the body

The body of the message may contain the following various types of data:

Table 3-2: Message Types

DATA TYPE	DESCRIPTION
char	1 byte (8 bits)
short	2 bytes, Little Endian Format
long	4 bytes, Little Endian Format

3.1. MESSAGE DEFINITIONS

The following messages have been defined:

Table 3-3: Messages

MESSAGE NAME	ID	ORIGINATOR	BODY CONTENT	DESCRIPTION
OPEN_DEVICE	1	Client	N/A	Initiates device opening
CLOSE_DEVICE	2	Client	N/A	Initiates device closing
START_DEVICE	3	Client	N/A	Initiates device start
STOP_DEVICE	4	Client	N/A	Initiates device stop
DATA	5	Server	Sampled Data	Data packet
DEVICE_OPENED	10	Server	-1	Reply to open, returns –1 if failed
DEVICE_CLOSED	20	Server	-1	Reply to open, returns –1 if failed
DEVICE_STARTED	30	Server	-1	Reply to open, returns –1 if failed
DEVICE_STOPPED	40	Server	-1	Reply to open, returns –1 if failed
STATUS	100	Client	N/A	Reserved
CONFIG_STATUS	101	Client	N/A	Reserved
DEVICE_STATUS_CONFIG	102	Client	N/A	Reserved
QUERY_STATUS	103	Client	N/A	Reserved

4. EMWIN LIBRARY

The EMWIN Library is the core piece of processing software. It performs the software demodulation based on the raw samples received from the Data Source Service. It also performs the format specific processing. A diagram showing the processing flow for both modulation types is below:

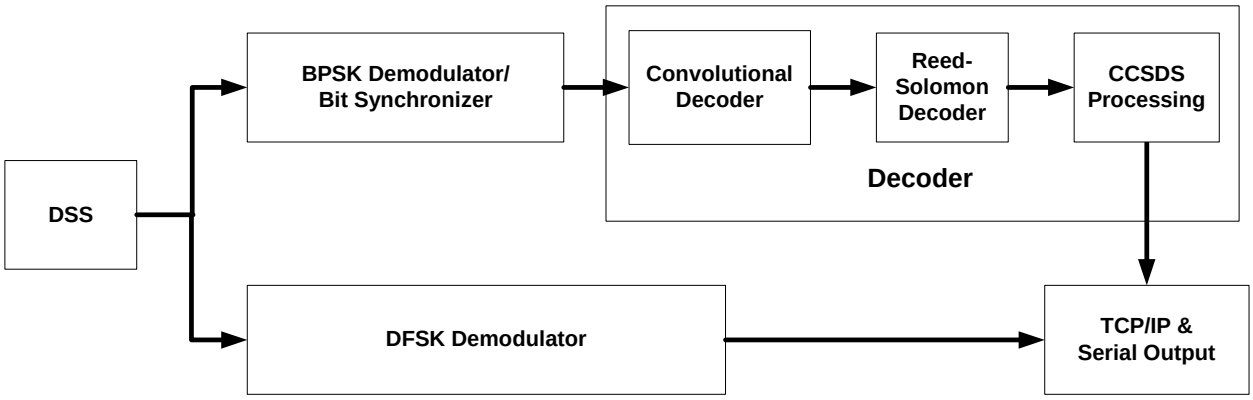

Figure 4-1: EMWIN Library Diagram

4.1. EMWIN-I (DFSK)

The EMWIN-I format provides the demodulation of the DFSK signal. At the time of this writing, this is the currently used system. This signal is currently broadcast from the GOES satellite on a frequency of 1690.725 MHz, which is 275 KHz lower than the standard 1691.0 MHz WEFAX signal. The DFSK signal operates at 9600 baud, and used a standard .35*baud FSK deviation, which results in the +/- 3360 Hz for the space and mark tones.

The EMWIN Library, after demodulating the received signal, outputs the data. In order to easily integrate with existing EMWIN software, it will output the data via the serial port, operating at 9600 baud, with NO parity, and 1 stop bit. It will also output the data as a TCP/IP server, listening on a port selected by the user.

4.2. EMWIN-N (BPSK)

The EMWIN-N format is the new signal format. It utilizes BPSK modulation, and the data is formatted according to CCSDS standards. It also utilizes concatenated convolutional coding with Reed-Solomon error detection and correction. The data rate after the convolutional encoding is approximately 22.08 Kbps.

After demodulating the data, performing the convolutional and Reed-Solomon decoding, as well and the CCSDS processing, it will output the data. The serial port for EMWIN-N operates at a higher baud rate, 19200 baud, with NO parity, and 1 stop bit. It will also output the data as a TCP/IP server, listening on a port selected by the user. The data output from both sources is the same as that for the EMWIN-I format, and can be directly input into existing EMWIN software.

4.3. PROGRAMMING INTERFACE

The programming interface provided for the EMWIN library consists of a set of C++ header files. These files declare the structures used to configure the library, as well as provide status for the various parts of

the processing. An export library for Microsoft Visual C++ 6.0 has also been provided for ease of integration into a user application.

The header files are provided in the *include/* directory, and includes the following files:

EMWINDLL.h ..... Declares the main interface class. Provides functions to configure the library, save the configuration, retrieve status, as well as start and stop processing.

EMWINConfig.h ..... Defines the main configuration structure. This structure holds all the configuration settings for the entire EMWIN processing chain.

DataSourceClientConfigStruct.h ..... Defines the structure used to configure the TCP/IP client that connects to the Data Source Service. Basically holds the IP number and port where the Data Source Service is listening.

FSKDemodConfigStruct.h ..... Defines the structure used to configure the FSK demodulator used for the EMWIN-I format.

BPSKDemodConfigStruct.h ..... Defines the structure used to configure the BPSK demodulator used for the EMWIN-N format.

SymbolTrackerConfigStruct.h ..... Defines the structure used to configure the symbol tracker that is used by the demodulator.

EMWINStatus.h ..... Defines the main status structure. This structure holds all the status items for the entire EMWIN processing chain.

BPSKReceiverStatusStruct.h ..... Defines the structure used to retrieve the status for the BSPK receiver used for EMWIN-N. Contains status for things such as the carrier frequency, symbol tracker status, and connection status of the Data Source Service.

FSKReceiverStatusStruct.h ..... Defines the structure used to retrieve the status for the FSK receiver used for EMWIN-I. Contains status for things such as the carrier frequency, symbol tracker status, and connection status of the Data Source Service.

EMWINBitProcessorStruct.h ..... Defines both the configuration and status structures for the post-demodulator processing. The configuration structure specifies the format, whether it is EMWIN-I or EMWIN-N, the IP and port on which the output server operates, as well as the serial port to output data on. The status structure gives such details such as the number of Reed-Solomon errors, as well as the number of bytes output for both the EMWIN-I and EMWIN-N formats.

SMsgHeader.h ..... Defines the message header structure used by the Data Source Service.

WaveBufferStruct.h ..... Defines the structure used for the samples retrieved from the Data Source Service.

WaveFormDebugStatusStruct.h ..... Defines the structure used to debug waveforms.

In addition, a sample application that interfaces with the library has been provided. The sample program, DLLTester, is a C++ application written using Microsoft Visual C++ 6.0. The appropriate workspace and project files are included with the DLLTester source code. The source code, DLLTester.cpp, shows the use of the library, from configuration, to starting the demodulator, and retrieving status.